Reconstrução da Chave Secreta do RSA Multi-primo

Reynaldo Cáceres Villena
(reynaldo@ime.usp.br)

Orientador: Routo Terada

Instituto de Matemática e Estatística - Universidade de São Paulo

Setembro de 2013

C A P E S

Agenda

1 - Objetivos

Objetivos

- Estudo e análise o algoritmo de reconstrução da chave secreta sk do criptossistema RSA básico ($u = 2$) apresentado por Heninger e Shacham.

Objetivos

- Estudo e análise o algoritmo de reconstrução da chave secreta sk do criptossistema RSA básico ($u = 2$) apresentado por Heninger e Shacham.
- Implementação de um algoritmo de reconstrução da chave secreta sk para o criptossistema RSA multi-primo ($u \geq 2$).

2 - Conceitos Básicos

RSA

- O RSA é o criptossistema de chave pública mais usado e implementado até a data.
- Seu nome é derivado das iniciais dos seus criadores: Ron (R)ivest, Adi (S)hamir e Len (A)dleman.
- Desde sua publicação, nenhum ataque conseguiu quebrá-lo, portanto não foi preciso mudar sua estrutura.
- Foi criado em agosto de 1977 no MIT[1] e publicado em fevereiro de 1978.

[1] Massachusetts Institute of Technology

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

1.-Algoritmo de Geração das chaves

$$N = \prod_{i=1}^{u} r_i \qquad\qquad ed = 1 \mod \phi(N)$$

- Chave pública $pk\langle N, e\rangle$ e chave secreta $sk\langle N, d\rangle$

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

1.-Algoritmo de Geração das chaves

$$N = \prod_{i=1}^{u} r_i \qquad\qquad ed = 1 \mod \phi(N)$$

- Chave pública $pk\langle N, e\rangle$ e chave secreta $sk\langle N, d\rangle$

2.- Algoritmo de encriptação

$M \in \mathbb{Z}_N$, $pk\langle N, e\rangle$

$$C = M^e \mod N$$

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

1.-Algoritmo de Geração das chaves

$$N = \prod_{i=1}^{u} r_i \qquad\qquad ed = 1 \mod \phi(N)$$

- Chave pública $pk\langle N, e\rangle$ e chave secreta $sk\langle N, d\rangle$

2.- Algoritmo de encriptação

$M \in \mathbb{Z}_N$, $pk\langle N, e\rangle$

$$C = M^e \mod N$$

3.- Algoritmo de decifração

C, $sk\langle N, d\rangle$

$$M = C^d \mod N$$

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

1.-Algoritmo de Geração das chaves

$$N = \prod_{i=1}^{u} r_i \qquad\qquad ed = 1 \mod \phi(N)$$

- Chave pública $pk\langle N, e\rangle$ e chave secreta $sk\langle N, d\rangle$

2.- Algoritmo de encriptação

$M \in \mathbb{Z}_N$, $pk\langle N, e\rangle$

$$C = M^e \mod N$$

3.- Algoritmo de decifração

C, $sk\langle N, d\rangle$

$$M = C^d \mod N$$

- **Criptossistema RSA Básico** $(u = 2)$
- **Criptossistema RSA Multi-primo** $(u \geq 3)$

Criptossistema RSA

O criptossistema RSA está conformado por 3 algoritmos

1.-Algoritmo de Geração das chaves

$$N = \prod_{i=1}^{u} r_i \qquad\qquad ed = 1 \mod \phi(N)$$

- Chave pública $pk\langle N, e\rangle$ e chave secreta $sk\langle N, d\rangle$

2.- Algoritmo de encriptação

$M \in \mathbb{Z}_N$, $pk\langle N, e\rangle$

$$C = M^e \mod N$$

3.- Algoritmo de decifração

C, $sk\langle N, d\rangle$

$$M = C^d \mod N$$

- **Criptossistema RSA Básico** $(u = 2)$
- **Criptossistema RSA Multi-primo** $(u \geq 3)$

O custo de execução do algoritmo de decifração é elevado.

QC-RSA (Decifração usando TCR)

O QC-RSA foi proposto por J-J. Quisquater and C. Couvreur em 1982. Esse método consiste em calcular M a partir de M_i usando o Teorema Chinês do Resto (TCR) onde

$$M_i = C^{d} \mod r_i \text{ para } 1 \leq i \leq u$$
$$M_i = C^{d_i} \mod r_i \text{ para } 1 \leq i \leq u$$

onde $d_i = d \mod (r_i - 1)$.

QC-RSA (Decifração usando TCR)

O QC-RSA foi proposto por J-J. Quisquater and C. Couvreur em 1982. Esse método consiste em calcular M a partir de M_i usando o Teorema Chinês do Resto (TCR) onde

$$M_i = C^d \mod r_i \text{ para } 1 \leq i \leq u$$
$$M_i = C^{d_i} \mod r_i \text{ para } 1 \leq i \leq u$$

onde $d_i = d \mod (r_i - 1)$.

Nova chave secreta sk

- $sk\langle r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$ com

$$r_2 r_2^{-1} = 1 \mod r_1$$
$$t_i(\prod_{j=1}^{i-1} r_j)^{-1} = 1 \mod r_i \text{ para } 3 \leq i \leq u$$

QC-RSA (Decifração usando TCR)

Algoritmo 1: Decifração-QC

Entrada: $sk\langle r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$, C

Saída: M

$M_1 = C^{d_1} \mod r_1$;

$M_2 = C^{d_2} \mod r_2$;

para $i = 3$ **to** u **faça**

  $M_i = C^{d_i} \mod r_i$;

$M = ((M_1 - M_2)r_2^{-1} \mod r_1)r_2 + M_2$;

$R = r_1$;

para $i = 3$ **to** u **faça**

  $R = R * r_{i-1}$;

  $M = (M_i - M)t^i \mod r_i)R + M$;

retorna M;

QC-RSA (Decifração usando TCR)

Algoritmo 2: Decifração-QC

Entrada: $sk\langle r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$, C
Saída: M

```
M_1 = C^{d_1} mod r_1;
M_2 = C^{d_2} mod r_2;
para i = 3 to u faça
    M_i = C^{d_i} mod r_i;
M = ((M_1 − M_2) r_2^{-1} mod r_1) r_2 + M_2;
R = r_1;
para i = 3 to u faça
    R = R ∗ r_{i−1};
    M = (M_i − M) t^i mod r_i) R + M;
retorna M;
```

O tempo de execução do QC-RSA:

- No criptossistema RSA básico é até 4 vezes mais rápido do que a execução de $M \leftarrow C^d \mod N$.
- No criptossistema RSA multi-primo é menor que do criptossistema RSA básico.

PKCS - Public Key Cryptography Standards

- O PKCS é um grupo de padrões devenvolvido pelos *laboratórios RSA*[2]
- O PKCS contém especificações para acelerar a implementação e desenvolvimento dos algoritmos dos criptossistemas de chave pública.

[2] Empresa dedicada à criptografia e ao software de segurança

PKCS - Public Key Cryptography Standards

- O PKCS é um grupo de padrões devenvolvido pelos *laboratórios RSA*[2]
- O PKCS contém especificações para acelerar a implementação e desenvolvimento dos algoritmos dos criptossistemas de chave pública.

PKCS #1

É o padrão e contém definições básicas e recomendações para a implementação do criptossistema RSA.

- Representação da chave pública
  1. $pk\langle N, e\rangle$ ($C = M^e \mod N$)
- Representações da chave secreta
  1. $sk\langle N, d\rangle$ ($M = C^d \mod N$).
  2. $sk\langle r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$ (QC-RSA).

[2] Empresa dedicada à criptografia e ao software de segurança

PKCS #1 - RSA (Recomendação para implementação do RSA)

Representação ANS.1 das chaves RSA segundo o padrão PKCS #1.

```
RSAPublicKey ::= SEQUENCE {
    modulus           INTEGER,  -- n
    publicExponent    INTEGER   -- e
}
RSAPrivateKey ::= SEQUENCE {
    version           Version,
    modulus           INTEGER,  -- n
    publicExponent    INTEGER,  -- e
    privateExponent   INTEGER,  -- d
    prime1            INTEGER,  -- p
    prime2            INTEGER,  -- q
    exponent1         INTEGER,  -- d mod (p-1)
    exponent2         INTEGER,  -- d mod (q-1)
    coefficient       INTEGER,  -- (inverse of q) mod p
    otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
Version ::= INTEGER { two-prime(0), multi(1) }
    (CONSTRAINED BY {-- version must be multi if otherPrimeInfos present --})
OtherPrimeInfos ::= SEQUENCE SIZE(1..MAX) OF OtherPrimeInfo
OtherPrimeInfo ::= SEQUENCE {
    prime             INTEGER,  -- ri
    exponent          INTEGER,  -- di
    coefficient       INTEGER   -- ti
}
```

- $sk\langle N, e, d, r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$.

Segurança do RSA

Requisito de um criptossistema de chave pública

A recuperação da chave secreta sk a partir da chave pública pk é um problema computacionalmente inviável.

Temos os valores $pk\langle N, e\rangle$, como achar o d?

$$Euclides-estendido(e, \phi(N)) \ll \phi(N) = \prod_{i=1}^{u}(r_i - 1) \ll Fatorar(N)$$

Segurança do RSA

Requisito de um criptossistema de chave pública

A recuperação da chave secreta sk a partir da chave pública pk é um problema computacionalmente inviável.

Temos os valores $pk\langle N, e\rangle$, como achar o d?

$$Euclides - estendido(e, \phi(N)) \ll \phi(N) = \prod_{i=1}^{u}(r_i - 1) \ll Fatorar(N)$$

Outros ataques como :

- O cálculo de $\phi(N)$ sem fatorar N
- A determinação de d sem fatorar N e sem calcular $\phi(N)$
- O cálculo de um d' equivalente a d

são mais difíceis que o $Fatorar(N)$

Segurança do RSA

Requisito de um criptossistema de chave pública

A recuperação da chave secreta sk a partir da chave pública pk é um problema computacionalmente inviável.

Temos os valores $pk\langle N, e\rangle$, como achar o d?

$$Euclides-estendido(e, \phi(N)) \ll \phi(N) = \prod_{i=1}^{u}(r_i - 1) \ll Fatorar(N)$$

Outros ataques como :

- O cálculo de $\phi(N)$ sem fatorar N
- A determinação de d sem fatorar N e sem calcular $\phi(N)$
- O cálculo de um d' equivalente a d

são mais difíceis que o $Fatorar(N)$

Segurança do RSA

Está baseado no problema de fatoração de inteiros (problema NP).

Algoritmos para fatorar inteiros

- O algoritmo NFS *(Number Field Sieve)* é o mais rápido para fatorar primos de mais de 100 dígitos. E seu tempo esperado é dado por

$$O(e^{1.923 \log^{\frac{1}{3}} n \log^{\frac{2}{3}} \log n}).$$

O maior inteiro fatorado foi de 232 dígitos (768 bits) (12-dez-2009).

Algoritmos para fatorar inteiros

- O algoritmo NFS *(Number Field Sieve)* é o mais rápido para fatorar primos de mais de 100 dígitos. E seu tempo esperado é dado por

  $$O(e^{1.923 \log^{\frac{1}{3}} n \log^{\frac{2}{3}} \log n}).$$

  O maior inteiro fatorado foi de 232 dígitos (768 bits) (12-dez-2009).
- O ECM *(Elliptic Curve Method)* usado na fatoração pode calcular um fator primo do inteiro N. Seu tempo é dado por

  $$O(\log^2 n e^{\sqrt{2} \log^{\frac{1}{2}} \frac{n}{u} \log^{\frac{1}{2}} \log \frac{n}{u}}).$$

  O maior fator encontrado foi de 75 dígitos (2-ago-2012).

Algoritmos para fatorar inteiros

- O algoritmo NFS *(Number Field Sieve)* é o mais rápido para fatorar primos de mais de 100 dígitos. E seu tempo esperado é dado por

  $$O(e^{1.923 \log^{\frac{1}{3}} n \log^{\frac{2}{3}} \log n}).$$

  O maior inteiro fatorado foi de 232 dígitos (768 bits) (12-dez-2009).
- O ECM *(Elliptic Curve Method)* usado na fatoração pode calcular um fator primo do inteiro N. Seu tempo é dado por

  $$O(\log^2 n e^{\sqrt{2} \log^{\frac{1}{2}} \frac{n}{u} \log^{\frac{1}{2}} \log \frac{n}{u}}).$$

  O maior fator encontrado foi de 75 dígitos (2-ago-2012).

O máximo valor estimado para u considerando que o inteiro N é seguro é dado por:

Número de bits (n)	1024	2048	3072	4096	8192
Máximo número de primes (u)	3	3	3	4	5

Fatoração do módulo N tendo informação extra

Com relação ao criptossistema RSA básico, temos que é possível fatorar o módulo $N = r_1 r_2$ em tempo polinomial tendo:

- os $n/4$ bits menos significativos ou mais significativos de r_1.
- os $n/4$ bits menos significativos de d.
- os $n/4$ bits menos significativos de d_1.
- 27% de bits aleatórios da chave secreta sk (**Algoritmo de reconstrução da chave secreta de Heninger e Shacham**).

Fatoração do módulo N tendo informação extra

Com relação ao criptossistema RSA básico, temos que é possível fatorar o módulo $N = r_1 r_2$ em tempo polinomial tendo:

- os $n/4$ bits menos significativos ou mais significativos de r_1.
- os $n/4$ bits menos significativos de d.
- os $n/4$ bits menos significativos de d_1.
- 27% de bits aleatórios da chave secreta sk (**Algoritmo de reconstrução da chave secreta de Heninger e Shacham**).

Com relação ao criptossistema RSA multi-primo, temos que é possível fatorar o módulo $N = \prod_{i=1}^{u} r_i$ em tempo polinomial tendo:

- os $\frac{ni}{u(i+1)}$ bits menos significativos ou mais significativos de r_i para $1 \leq i \leq u - 1$.

3 - Algoritmo de Reconstrução da Chave Secreta RSA

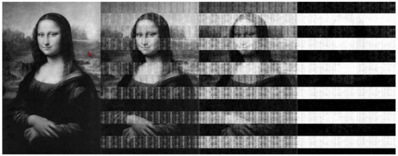
Ataques *cold-boot*

cold-boot ou *hard-boot*

Se refere ao *boot* de um computador depois de ter sido cortada sua fonte de energia abruptamente.

Remanescência da memória DRAM

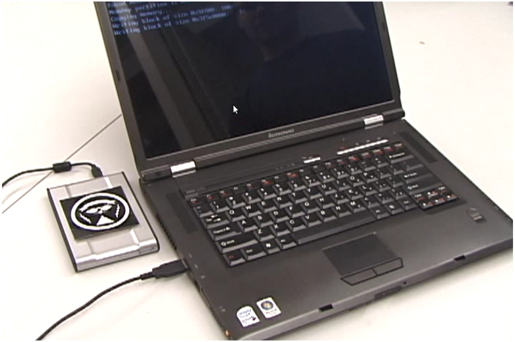
Capacidade de conservação de dados da memória DRAM/SRAM depois de se aplicar um *cold-boot*.



5s. 30s. 1m. 5m.

Um ataque *cold-boot* é um tipo de ataque onde o atacante utiliza dados que foram obtidos da memória DRAM/SRAM depois de aplicar um *cold-boot* ao computador.

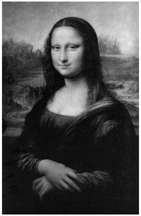
Gerando imagem da memória DRAM



Iniciando um programa *Syslinux bootloader* para obter uma imagem da memória DRAM.

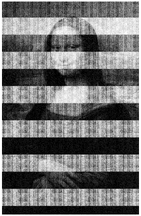
Definição de $\tilde{sk}$

sk



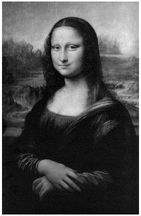
Ataque
Cold Boot
$\rightarrow$

$\tilde{sk}(\delta)$



Algoritmo de
reconstrução
$\rightarrow$

sk



Onde $\tilde{sk}$ contém uma porcentagem δ de bits corretos de sk.

Relações matemáticas da chave secreta

Temos a chave secreta

$$sk\langle N, e, d, r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$$

segundo o padrão PKCS # 1, onde suas relações matemáticas são dados por:

$$\begin{aligned}
N &= \prod_{i=1}^{u} r_i & e.d &= 1 \mod \prod_{i=1}^{u}(r_i - 1)\\
e.d_1 &= 1 \mod (r_1 - 1) & r_2.r_2^{-1} &= 1 \mod r_1\\
e.d_2 &= 1 \mod (r_2 - 1) & r_1.r_2.t_3 &= 1 \mod r_3\\
&\vdots & &\vdots\\
e.d_u &= 1 \mod (r_u - 1) & r_1.r_2...r_{u-1}.t_u &= 1 \mod r_u
\end{aligned}$$

Relações matemáticas da chave secreta

Temos a chave secreta

$$sk\langle N, e, d, r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$$

segundo o padrão PKCS # 1, onde suas relações matemáticas são dados por:

$$N = \prod_{i=1}^{u} r_i \qquad e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$e.d_1 = 1 + k_1(r_1 - 1) \qquad r_2.r_2^{-1} = 1 + g.r_1$$

$$e.d_2 = 1 + k_2(r_2 - 1) \qquad r_1.r_2.t_3 = 1 + g_3.r_3$$

$$\vdots \qquad \vdots$$

$$e.d_u = 1 + k_u(r_u - 1) \qquad r_1.r_2...r_{u-1}.t_u = 1 + g_u.r_u$$

Relações matemáticas da chave secreta

Temos a chave secreta

$$sk\langle N, e, d, r_1, r_2, d_1, d_2, r_2^{-1}, \langle r_3, d_3, t_3\rangle, .., \langle r_u, d_u, t_u\rangle\rangle$$

segundo o padrão PKCS # 1, onde suas relações matemáticas são dados por:

$$N = \prod_{i=1}^{u} r_i \qquad e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$\begin{aligned} e.d_1 &= 1 + k_1(r_1 - 1) \\ e.d_2 &= 1 + k_2(r_2 - 1) \\ &\vdots \\ e.d_u &= 1 + k_u(r_u - 1) \end{aligned}$$

Calculando os valores de $k, k_1, ...k_u$

Analisando os valores $k, k_1, ...k_u$.

$$N = \prod_{i=1}^{u} r_i$$

$$e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$e.d_i = 1 + k_i(r_i - 1) \text{ para } 1 \leq i \leq u$$

$$e, d \in \mathbb{Z}^*_{\phi(N)} \Rightarrow e, d < \phi(N) \Rightarrow k < e$$

$$e, d_i \in \mathbb{Z}^*_{\phi(r_i)} \Rightarrow e, d < \phi(r_i) \Rightarrow k_i < e$$

$$0 \leq k, k_1, ..., k_u < e$$

Calculando os valores de $k, k_1, ...k_u$

Analisando os valores $k, k_1, ...k_u$.

$$N = \prod_{i=1}^{u} r_i$$

$$e.d = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$e.d_i = 1 + k_i(r_i - 1) \text{ para } 1 \leq i \leq u$$

$$e, d \in \mathbb{Z}^*_{\phi(N)} \Rightarrow e, d < \phi(N) \Rightarrow k < e$$

$$e, d_i \in \mathbb{Z}^*_{\phi(r_i)} \Rightarrow e, d < \phi(r_i) \Rightarrow k_i < e$$

$$\boxed{0 \leq k, k_1, ..., k_u < e}$$

Aplicando $\mod e$ para calcular os valores $k, k_1, ...k_u$.

$$N = \prod_{i=1}^{u} r_i$$

$$0 = 1 + k\prod_{i=1}^{u}(r_i - 1)$$

$$0 = 1 + k_i(r_i - 1) \text{ para } 1 \leq i \leq u$$

$$\boxed{N\prod_{i=1}^{u} k_i = \prod_{i=1}^{u}(k_i - 1)}$$

$$\boxed{\prod_{i=1}^{u} k_i = k(-1)^{u-1}}$$

Calculando os valores de $k, k_1, ...k_u$

$$N \prod_{i=1}^{u} k_i = \prod_{i=1}^{u} (k_i - 1)$$

$$\prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

Onde o número de soluções possíveis para $\langle k, k_1, ..., k_u \rangle$:

$$\alpha(u) = \left\{ \begin{array}{ll} 0 & \text{se } u = 1 \text{ e } N \equiv 1 \mod e \\ 1 & \text{se } u = 1 \text{ e } N \not\equiv 1 \mod e \\ (e-2)^{u-1} - \alpha(u-1) & \text{se } u > 1 \end{array} \right\}$$

Calculando os valores de $k, k_1, \ldots k_u$

$$N \prod_{i=1}^{u} k_i = \prod_{i=1}^{u} (k_i - 1)$$

$$\prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

Onde o número de soluções possíveis para $\langle k, k_1, \ldots, k_u \rangle$:

$$\alpha(u) = \begin{cases} 0 & \text{se } u = 1 \text{ e } N \equiv 1 \mod e \\ 1 & \text{se } u = 1 \text{ e } N \not\equiv 1 \mod e \\ (e-2)^{u-1} - \alpha(u-1) & \text{se } u > 1 \end{cases}$$

Para o criptossistema RSA básico ($u = 2$)

Percival formulou que existe e possíveis soluções para $\langle k, k_1, k_2 \rangle$.

- É possível encontrar melhores valores para o k?

Calculando melhores valores para k

Lema:

Para um e pequeno, os $\frac{n}{u}$ bits mais significativos de d pode ser estimados eficientemente.

$$e.d = 1 + k\prod_{i=1}^{u}(r_i - 1) + kN - k\prod_{i=1}^{u} r_i$$

$$d = \frac{1 + kN}{e} + \frac{k}{e}\left(\prod_{i=1}^{u}(r_i - 1) - \prod_{i=1}^{u} r_i\right)$$

$$d = d_0 + d_1 \qquad \text{onde } lg|d_1| \approx n - \frac{n}{u}$$

Calculando melhores valores para k

Lema:

Para um e pequeno, os $\frac{n}{u}$ bits mais significativos de d pode ser estimados eficientemente.

$$e.d = 1 + k\prod_{i=1}^{u}(r_i - 1) + kN - k\prod_{i=1}^{u} r_i$$

$$d = \frac{1+kN}{e} + \frac{k}{e}\left(\prod_{i=1}^{u}(r_i - 1) - \prod_{i=1}^{u} r_i\right)$$

$$d = d_0 + d_1 \qquad \text{onde } lg|d_1| \approx n - \frac{n}{u}$$

Em outras palavras, se calculamos o valor de

$$d_0(k') = \frac{1+k'N}{e}$$

com o valor correto de $k' = k$, vamos ter em $d_0(k')$ os $\frac{n}{u}$ bits mais significativos de d.

Calculando melhores valores para k

Lembramos que temos $\tilde{sk}$ com uma porcentagem δ de bits corretos

- Temos em $\tilde{d}$ um total de δn bits corretos

Calculamos o valor de d_0 para cada valor possível de $k'(0 < k < e)$.

$$d_0(k') = \frac{1 + k'N}{e} \text{ para } 0 < k < e$$

onde um dos $d_0(k')$ tem o bits $\frac{n}{u}$ mais significativos de d.

[3] Número de bits diferentes entre duas variáveis

Calculando melhores valores para k

Lembramos que temos $\tilde{sk}$ com uma porcentagem δ de bits corretos

- Temos em $\tilde{d}$ um total de δn bits corretos

Calculamos o valor de d_0 para cada valor possível de $k'(0 < k < e)$.

$$d_0(k') = \frac{1 + k'N}{e} \text{ para } 0 < k < e$$

onde um dos $d_0(k')$ tem o bits $\frac{n}{u}$ mais significativos de d.

- Os melhores valores para k estão dados quando o $d_0(k')$ e $\tilde{d}$ tenham a menor distância de *Hamming*[3]

$$H(k') = \sum_{i=n-\frac{n}{u}}^{n} d_0(k')[i] \oplus \tilde{d}[i]$$

Melhor valor para k

$$k = \{k/H(k) = \min(H(1), H(2), ..., H(e-1))\}$$

[3] Número de bits diferentes entre duas variáveis

Calculando $\langle k_1, ..., k_u \rangle$ onde k é conhecido

Determinando os valores de $\langle k_1, ..., k_u \rangle$ a partir de k:

$$N \prod_{i=1}^{u} k_i = \prod_{i=1}^{u} (k_i - 1)$$

$$\prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

Vamos supor que temos $L \equiv \prod_{i=1}^{u-2} [1 - (k_i)^{-1}]$ e $M \equiv \prod_{i=1}^{u-2} k_i$. Portanto

$$0 \equiv k_{u-1}^2 - [(-1)^u k M^{-1} [N L^{-1} - 1] + 1] k_{u-1} - (-1)^u k M^{-1}$$

Onde o número de soluções possíveis para $\langle k_1, ..., k_u \rangle$ conhecendo k:

$$\boxed{\beta(u) \leq 2(e-2)^{u-2}}$$

Calculando $\langle k_1, ..., k_u \rangle$ onde k é conhecido

Determinando os valores de $\langle k_1, ..., k_u \rangle$ a partir de k:

$$N \prod_{i=1}^{u} k_i = \prod_{i=1}^{u} (k_i - 1)$$

$$\prod_{i=1}^{u} k_i = k(-1)^{u-1}$$

Vamos supor que temos $L \equiv \prod_{i=1}^{u-2} [1 - (k_i)^{-1}]$ e $M \equiv \prod_{i=1}^{u-2} k_i$. Portanto

$$0 \equiv k_{u-1}^2 - [(-1)^u k M^{-1} [N L^{-1} - 1] + 1] k_{u-1} - (-1)^u k M^{-1}$$

Onde o número de soluções possíveis para $\langle k_1, ..., k_u \rangle$ conhecendo k:

$$\boxed{\beta(u) \leq 2(e - 2)^{u-2}}$$

Para o criptossistema RSA básico $(u = 2)$

Heninger mostrou o seguinte sistema de equações para $\langle k, k_1, k_2 \rangle$:

$$0 = k_2^2 - [k(N - 1) + 1] k_2 - k$$
$$0 = k + k_1 k_2$$

Correção dos bits inferiores de:

Temos $\tilde{sk}\langle \tilde{d}, \tilde{r_1}, \tilde{r_2}, \tilde{d_1}, \tilde{d_2}, \langle \tilde{r_3}, \tilde{d_3} \rangle, ..., \langle \tilde{r_u}, \tilde{d_u} \rangle \rangle$

- Sabemos que r_is são primos portanto podemos corrigir

$$\tilde{r_i}[0] = 1 \text{ para } 1 \leq i \leq u.$$

Correção dos bits inferiores de:

Temos $\tilde{sk}\langle \tilde{d}, \tilde{r_1}, \tilde{r_2}, \tilde{d_1}, \tilde{d_2}, \langle \tilde{r_3}, \tilde{d_3}\rangle, ..., \langle \tilde{r_u}, \tilde{d_u}\rangle\rangle$

- Sabemos que r_is são primos portanto podemos corrigir

$$\tilde{r_i}[0] = 1 \text{ para } 1 \leq i \leq u.$$

Seja $\tau(x)$: o maior expoente da potência de 2 tal que $2^{\tau(x)}|x$.

- Sabemos que $2|r_i - 1$, então temos que $2^{1+\tau(k_i)}|k_i(r_i - 1)$

$$2^{1+\tau(k_i)}|ed_i - 1$$

$$ed_i - 1 \equiv 0 \mod 2^{1+\tau(k_i)}$$
$$d_i \equiv e^{-1} \mod 2^{1+\tau(k_i)} \qquad \text{para } 1 \leq i \leq u$$

portanto podemos corrigir

$$\tilde{d_i}[j] = (e^{-1} \mod 2^{1+\tau(k_i)})[j] \text{ para } 0 \leq j \leq 1 + \tau(k_i) \text{ e } 1 \leq i \leq u$$

Correção dos bits inferiores de:

- Sabemos que $2|r_i - 1$, então temos que $2^u|\prod_{i=1}^{u}(r_i - 1)$

$$2^{u+\tau(k)}|k\prod_{i=1}^{u}(r_i - 1)$$

$$2^{u+\tau(k)}|ed - 1$$

$$ed - 1 \equiv 0 \mod 2^{u+\tau(k)}$$

$$d \equiv e^{-1} \mod 2^{u+\tau(k)} \qquad \text{para } 1 \leq i \leq u$$

portanto podemos corrigir

$$d[j] \equiv (e^{-1} \mod 2^{u+\tau(k)})[j] \text{ para } 0 \leq j \leq u + \tau(k)$$

Ideia do Algoritmo de Reconstrução

Temos as equações do criptossistema RSA

$$N = \prod_{i=1}^{u} r_i$$

$$ed = 1 + k \prod_{i=1}^{u} (r_i - 1)$$

$$ed_i = 1 + k_i(r_i - 1) \qquad \text{para } 1 \leq i \leq u$$

Ideia do Algoritmo de Reconstrução

Equações RSA definidas como polinômios

$$f_1(x_1, x_2, .., x_u) = N - \prod_{i=1}^{u} x_i$$

$$f_2(x_1, x_2, ...x_u, y) = ey - 1 - k \prod_{i=1}^{u}(x_i - 1)$$

$$f_{3_i}(x_i, y_i) = ey_i - 1 - k_i(x_i - 1) \qquad \text{para } 1 \leq i \leq u$$

onde a raiz solução desses polinômios é dado pela chave secreta sk

$$(y, x_1, x_2, y_1, y_2, (x_3, y_3), ..., (x_u, y_u)) = sk\langle d, r_1, r_2, d_1, d_2, \langle r_3, d_3\rangle, ..., \langle r_u, d_u\rangle\rangle$$

Lema de Hensel

Lema de Hensel para multivariáveis

Uma raiz $r = (r_1, r_2, ..., r_u)$ do polinômio $f(x_1, x_2, ..., x_u) \mod \pi^j$ pode ser usada para gerar uma raiz $r + b \mod \pi^{j+1}$ se $b = (b_1\pi^j, b_2\pi^j, ..., b_u\pi^j)$, $0 \leq b_i \leq \pi - 1$ é uma solução para a equação

$$f(r+b) = f(r) + \textstyle\sum_i b_i \pi^j f_{x_i}(r) \equiv 0 \pmod{\pi^{j+1}}$$

(onde, f_{x_j} é a derivada parcial de f com relação a x_j)

Onde a partir de uma raiz

$$\langle d', r'_1, r'_2, d'_1, d'_2, \langle r'_3, d'_3\rangle, ..., \langle r'_u, d'_u\rangle\rangle \,|\, \begin{array}{l} f_1(r'_1, r'_2, .., r'_u) \mod 2^j \\ f_2(r'_1, r'_2, .., r'_u, d') \mod 2^{j+\tau(k)} \\ f_{3_i}(r'_i, d'_i) \mod 2^{j+\tau(k_i)} \text{ para } 1 \leq i \leq u \end{array}$$

Lema de Hensel

Lema de Hensel para multivariáveis

Uma raiz $r = (r_1, r_2, ..., r_u)$ do polinômio $f(x_1, x_2, ..., x_u) \mod \pi^j$ pode ser usada para gerar uma raiz $r + b \mod \pi^{j+1}$ se $b = (b_1\pi^j, b_2\pi^j, ..., b_u\pi^j)$, $0 \leq b_i \leq \pi - 1$ é uma solução para a equação

$$f(r+b) = f(r) + \sum_i b_i \pi^j f_{x_i}(r) \equiv 0 \pmod{\pi^{j+1}}$$

(onde, f_{x_j} é a derivada parcial de f com relação a x_j)

vamos gerar uma raiz

$$\langle d^*, r_1^*, r_2^*, d_1^*, d_2^*, \langle r_3^*, d_3^* \rangle, ..., \langle r_u^*, d_u^* \rangle \rangle \; | f_1(r_1^*, r_2^*, .., r_u^*) \mod 2^{1+j}$$

$$f_2(r_1^*, r_2^*, .., r_u^*, d^*) \mod 2^{1+j+\tau(k)}$$

$$f_{3_i}(r_i^*, d_i^*) \mod 2^{1+j+\tau(k_i)} \text{ para } 1 \leq i \leq u.$$

Lema de Hensel

Lema de Hensel para multivariáveis

Uma raiz $r = (r_1, r_2, ..., r_u)$ do polinômio $f(x_1, x_2, ..., x_u) \mod \pi^j$ pode ser usada para gerar uma raiz $r + b \mod \pi^{j+1}$ se $b = (b_1\pi^j, b_2\pi^j, ..., b_u\pi^j)$, $0 \leq b_i \leq \pi - 1$ é uma solução para a equação

$$f(r+b) = f(r) + \sum_i b_i \pi^j f_{x_i}(r) \equiv 0 \pmod{\pi^{j+1}}$$

(onde, f_{x_j} é a derivada parcial de f com relação a x_j)

Segundo o lema de Hensel

$$\begin{aligned} d^* &= d' + 2^{j+\tau(k)} d[j+\tau(k)] \\ r_i^* &= r_i' + 2^j r_i[j] && \text{para } 1 \leq i \leq u \\ d_i^* &= d_i' + 2^{j+\tau(k_i)} d_i[j+\tau(k_i)] && \text{para } 1 \leq i \leq u \end{aligned}$$

Lema de Hensel

Lema de Hensel para multivariáveis

Uma raiz $r = (r_1, r_2, ..., r_u)$ do polinômio $f(x_1, x_2, ..., x_u) \mod \pi^j$ pode ser usada para gerar uma raiz $r + b \mod \pi^{j+1}$ se $b = (b_1\pi^j, b_2\pi^j, ..., b_u\pi^j)$, $0 \leq b_i \leq \pi - 1$ é uma solução para a equação

$$f(r+b) = f(r) + \sum_i b_i \pi^j f_{x_i}(r) \equiv 0 \pmod{\pi^{j+1}}$$

(onde, f_{x_j} é a derivada parcial de f com relação a x_j)

Onde devem se cumprir as seguintes equivalências:

$$\left(N - \prod_{i=1}^{u} r_i'\right)[j] \equiv \sum_{i=1}^{u} r_i[j] \mod 2$$

$$\left(ed' - 1 - k\prod_{i=1}^{u}(r_i' - 1)\right)[j + \tau(k)] \equiv d[j + \tau(k)] \mod 2$$

$$(ed_i - 1 - k_i(r_i' - 1))[j + \tau(k_i)] \equiv r_i[j] + d_i[j + \tau(k_i)] \mod 2 \text{ para } 1 \leq i \leq u$$

Algoritmo de Reconstrução

Vamos definir $root[j-1]$ como um conjunto de raízes do tipo

$$\langle d', r'_1, r'_2, d'_1, d'_2, \langle r'_3, d'_3 \rangle, ..., \langle r'_u, d'_u \rangle \rangle$$

que é raiz comum dos polinômios RSA

$$f_1(r'_1, r'_2, .., r'_u) \mod 2^j$$

$$f_2(r'_1, r'_2, .., r'_u, d') \mod 2^{j+\tau(k)}$$

$$f_{3_i}(r'_i, d'_i) \mod 2^{j+\tau(k_i)} \text{ para } 1 \leq i \leq u$$

Algoritmo de Reconstrução

Vamos definir $root[j-1]$ como um conjunto de raízes do tipo

$$\langle d', r'_1, r'_2, d'_1, d'_2, \langle r'_3, d'_3\rangle, ..., \langle r'_u, d'_u\rangle\rangle$$

que é raiz comum dos polinômios RSA

$$f_1(r'_1, r'_2, .., r'_u) \mod 2^j$$

$$f_2(r'_1, r'_2, .., r'_u, d') \mod 2^{j+\tau(k)}$$

$$f_{3_i}(r'_i, d'_i) \mod 2^{j+\tau(k_i)} \text{ para } 1 \leq i \leq u$$

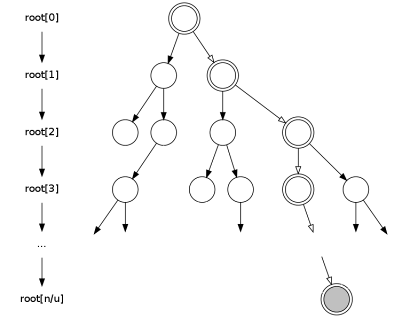
Algoritmo de Reconstrução

$$root[0] \rightarrow root[1] \rightarrow root[2] \rightarrow ... \rightarrow root\left[\frac{n}{u}\right]$$

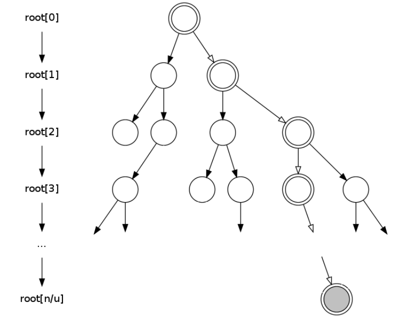
onde:

- $root[0] = [\langle e^{-1}, 1_1, 1_2, e_1^{-1}, e_2^{-1}, \langle 1_3, e_3^{-1}\rangle, ..., \langle 1_u, e_u^{-1}\rangle\rangle]$ com $e^{-1} = e^{-1} \mod 2^{1+\tau(k)}$ e $e_i^{-1} = e^{-1} \mod 2^{1+\tau(k_i)}$ para $1 \leq i \leq u$.
- $\langle \tilde{d}, r_1, r_2, d_1, d_2, \langle r_3, d_3\rangle, ..., \langle r_u, d_u\rangle\rangle \in root\left[\frac{n}{u}\right]$

Complexidade do Algoritmo de Reconstrução



Complexidade do Algoritmo de Reconstrução



Análise da Complexidade do Algoritmo de Fatoração do Inteiro

- G: Número de raízes incorretas geradas por uma raiz boa.
- B: Número de raízes incorretas geradas por uma raiz incorreta.
- X_j: Número de raízes incorretas geradas no nível j ($|root[j]|$).
- Função de recorrência: $X_j = X_{j-1}B + G$

Número de raízes geradas por uma raiz boa

- Seja $\langle d', r_1', r_2', d_1', d_2', \langle r_3', d_3' \rangle, ..., \langle r_u', d_u' \rangle \rangle$ a raiz boa de $root[j-1]$
- Temos uma porcentagem δ de bits conhecidos em $\tilde{sk}$

$$\left(N - \prod_{i=1}^{u} r_i'\right)[j] \equiv c_1 \equiv \sum_{i=1}^{u} r_i[j] \mod 2$$

$$\left(ed' - 1 - k\prod_{i=1}^{u}(r_i' - 1)\right)[j + \tau(k)] \equiv c_2 \equiv d[j + \tau(k)] \mod 2$$

$$(ed_i - 1 - k_i(r_i' - 1))[j + \tau(k_i)] \equiv c_{3_i} \equiv r_i[j] + d_i[j + \tau(k_i)] \mod 2$$

para $1 \leq i \leq u$

Número de raízes geradas por uma raiz boa

- Seja $\langle d', r'_1, r'_2, d'_1, d'_2, \langle r'_3, d'_3\rangle, ..., \langle r'_u, d'_u\rangle\rangle$ a raiz boa de $root[j-1]$
- Temos uma porcentagem δ de bits conhecidos em $\tilde{sk}$

$$\left(N - \prod_{i=1}^{u} r'_i\right)[j] \equiv c_1 \equiv \sum_{i=1}^{u} r_i[j] \mod 2$$

$$\left(ed' - 1 - k\prod_{i=1}^{u}(r'_i - 1)\right)[j + \tau(k)] \equiv c_2 \equiv d[j + \tau(k)] \mod 2$$

$$(ed_i - 1 - k_i(r'_i - 1))[j + \tau(k_i)] \equiv c_{3_i} \equiv r_i[j] + d_i[j + \tau(k_i)] \mod 2$$

para $1 \leq i \leq u$

Número de raízes incorretas geradas por uma raiz boa

$$\mathbb{E}[G] = \sum_{h=1}^{u} (2^{h-1} - 1)\binom{u}{h}(1-\delta)^{2h}(2\delta(1-\delta) + \delta^2)^{u-h}$$

Número de raízes geradas por uma raiz incorreta

- Seja $\langle d', r'_1, r'_2, d'_1, d'_2, \langle r'_3, d'_3\rangle, \ldots, \langle r'_u, d'_u\rangle\rangle$ a raiz incorreta de $root[j-1]$
- Temos uma porcentagem δ de bits conhecidos em $\tilde{sk}$

$$\left(N - \prod_{i=1}^{u} r'_i\right)[j] \equiv \{c_1, \overline{c_1}\} \equiv \sum_{i=1}^{u} r_i[j] \mod 2$$

$$\left(ed' - 1 - k\prod_{i=1}^{u}(r'_i - 1)\right)[j+\tau(k)] \equiv \{c_2, \overline{c_2}\} \equiv d[j+\tau(k)] \mod 2$$

$$(ed_i - 1 - k_i(r'_i - 1))[j+\tau(k_i)] \equiv \{c_{3_i}, \overline{c_{3_i}}\} \equiv r_i[j] + d_i[j+\tau(k_i)] \mod 2$$

para $1 \leq i \leq u$

Número de raízes geradas por uma raiz incorreta

- Seja $\langle d', r'_1, r'_2, d'_1, d'_2, \langle r'_3, d'_3 \rangle, \ldots, \langle r'_u, d'_u \rangle \rangle$ a raiz incorreta de $root[j-1]$
- Temos uma porcentagem δ de bits conhecidos em $\tilde{sk}$

$$\left(N - \prod_{i=1}^{u} r'_i\right)[j] \equiv \{c_1, \overline{c_1}\} \equiv \sum_{i=1}^{u} r_i[j] \mod 2$$

$$\left(ed' - 1 - k\prod_{i=1}^{u}(r'_i - 1)\right)[j+\tau(k)] \equiv \{c_2, \overline{c_2}\} \equiv d[j+\tau(k)] \mod 2$$

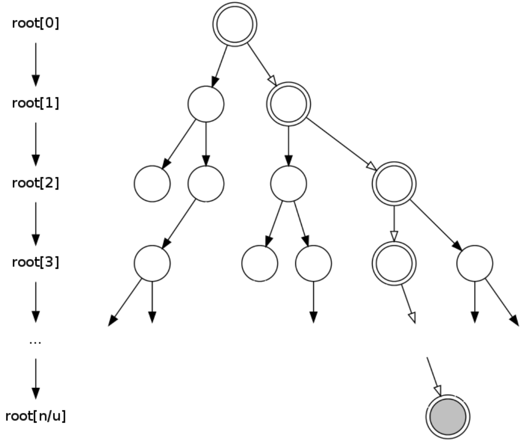
$$(ed_i - 1 - k_i(r'_i - 1))[j+\tau(k_i)] \equiv \{c_{3_i}, \overline{c_{3_i}}\} \equiv r_i[j] + d_i[j+\tau(k_i)] \mod 2$$

para $1 \le i \le u$

Número de raízes incorretas geradas por uma raiz incorreta

$$\mathbb{E}[B] = \frac{(2-\delta)^{2u+1}}{2^{u+2}}$$

Número de Soluções Incorretas Geradas no nível j



Função de Recorrência:

$$\mathbb{E}[X_j] = \mathbb{E}[X_{j-1}]\mathbb{E}[B] + \mathbb{E}[G], \text{ com } \mathbb{E}[X_1] = \mathbb{E}[G].$$

Número de Raízes Incorretas Geradas no nível j

Função de Recorrência: $\mathbb{E}[X_j] = \mathbb{E}[X_{j-1}]\mathbb{E}[B] + \mathbb{E}[G]$ com $\mathbb{E}[X_1] = \mathbb{E}[G]$

$$\mathbb{E}[X_j] = \mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]}$$

$$\begin{aligned}\mathbb{V}ar[X_j] =& \mathbb{E}[B]^{2(j-1)}\left[\frac{\mathbb{E}[G](\mathbb{E}[B^2]-\mathbb{E}[B]+\mathbb{E}[B]\mathbb{E}[G])\mathbb{E}[B]}{(1-\mathbb{E}[B])(1-\mathbb{E}[B]^2)}\right] + \mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]} \\ & - \mathbb{E}[B]^{j-1}\left[\frac{\mathbb{E}[G][\mathbb{E}[B^2]-\mathbb{E}[B]+2\mathbb{E}[B]\mathbb{E}[G]]}{(1-\mathbb{E}[B])^2}\right] - \left[\mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]}\right]^2 \\ & \frac{1}{1-\mathbb{E}[B]^2}\left[\frac{\mathbb{E}[G][\mathbb{E}[B^2]-\mathbb{E}[B]+2\mathbb{E}[B]\mathbb{E}[G]]}{1-\mathbb{E}[B]}\right]\end{aligned}$$

Onde $\mathbb{E}[X_j]$ e $\mathbb{V}ar[X_j]$ são funções exponenciais.

Número de raízes incorretas analisadas na execução de algoritmo

O algoritmo é executado para $1 \leq j \leq \frac{n}{u}$.

$$\mathbb{E}\left[\sum_{j=1}^{\frac{n}{u}} X_j\right] = \sum_{j=1}^{\frac{n}{u}} \mathbb{E}\left[X_j\right] = \sum_{j=1}^{\frac{n}{u}} \mathbb{E}[G]\frac{1-\mathbb{E}[B]^j}{1-\mathbb{E}[B]}$$

$$= \frac{n}{u}\frac{\mathbb{E}[G]}{1-\mathbb{E}[B]} + \frac{\mathbb{E}[G]\mathbb{E}[B](\mathbb{E}[B]^{\frac{n}{u}}-1)}{(\mathbb{E}[B]-1)^2}$$

$$\mathbb{V}ar\left[\sum_{j=1}^{\frac{n}{u}} X_j\right] = \sum_{l=1}^{\frac{n}{u}}\sum_{j=1}^{\frac{n}{u}} Cov(X_l, X_j) \leq \sum_{l=1}^{\frac{n}{u}}\sum_{j=1}^{\frac{n}{u}} \sqrt{\mathbb{V}ar[X_l]\mathbb{V}ar[X_j]}$$

$$\leq \sum_{l=1}^{\frac{n}{u}}\sum_{j=1}^{\frac{n}{u}} \sqrt{\max(\mathbb{V}ar[X_1], .., \mathbb{V}ar[X_{\frac{n}{u}}])^2}$$

$$\leq \left(\frac{n}{u}\right)^2 \max(\mathbb{V}ar[X_1], .., \mathbb{V}ar[X_{\frac{n}{u}}])$$

Número de raízes incorretas analisadas na execução de algoritmo

Onde o comportamento de $\mathbb{E}\left[\sum_{j=1}^{\frac{n}{u}} X_j\right]$ e $\mathbb{V}ar\left[\sum_{j=1}^{\frac{n}{u}} X_j\right]$ podem ser:

- Exponencial ($\mathbb{E}[B] > 1$ já que $\lim_{n\to\infty} \mathbb{E}[B]^{\frac{n}{u}} = +\infty$)
- Polinomial ($\mathbb{E}[B] < 1$ já que $\lim_{n\to\infty} \mathbb{E}[B]^{\frac{n}{u}} = 0 < 1$)

Número de raízes incorretas analisadas na execução de algoritmo

Onde o comportamento de $\mathbb{E}\left[\sum_{j=1}^{\frac{n}{u}} X_j\right]$ e $\mathbb{V}ar\left[\sum_{j=1}^{\frac{n}{u}} X_j\right]$ podem ser:

- Exponencial ($\mathbb{E}[B] > 1$ já que $\lim_{n\to\infty} \mathbb{E}[B]^{\frac{n}{u}} = +\infty$)
- Polinomial ($\mathbb{E}[B] < 1$ já que $\lim_{n\to\infty} \mathbb{E}[B]^{\frac{n}{u}} = 0 < 1$)

Onde para $\mathbb{E}[B] < 1$ vamos obter:

$$\mathbb{E}\left[\sum_{j=1}^{\frac{n}{u}} X_j\right] = \frac{n}{u}\frac{\mathbb{E}[G]}{1-\mathbb{E}[B]} + \frac{\mathbb{E}[G]\mathbb{E}[B](\mathbb{E}[B]^{\frac{n}{u}}-1)}{(\mathbb{E}[B]-1)^2} < \frac{n}{u}\frac{\mathbb{E}[G]}{1-\mathbb{E}[B]}$$

$$\mathbb{V}ar\left[\sum_{j=1}^{\frac{n}{u}} X_j\right] \leq \left(\frac{n}{u}\right)^2 \max(\mathbb{V}ar[X_1], .., \mathbb{V}ar[X_{\frac{n}{u}}]),$$

onde $\mathbb{V}ar[X_j]$ agora é polinomial.

Complexidade do Algoritmo de Reconstrução

Teorema de Chebyshev

A desigualdade de Chebyshev proporciona a probabilidade de obter que o valor da v.a. fique longe de certo número de vezes do desvio padrão em relação ao valor esperado.

$$P(\mathbb{E}[X] - c\sigma < X < \mathbb{E}[X] + c\sigma) \geq 1 - \frac{1}{c^2}$$

A probabilidade de que qualquer v.a. tenha um valor dentro das c desviações estândar do valor esperado é pelo menos de $1 - \frac{1}{c^2}$.

Onde aplicado ao problema de reconstrução da chave secreta RSA, temos que a probabilidade de analisar mais de

$$\mathbb{E}[\sum_{j=1}^{\frac{n}{u}} X_j] + n\sqrt{\mathbb{V}ar[\sum_{j=1}^{\frac{n}{u}} X_j]} \leq \frac{n}{u}\frac{\mathbb{E}[G]}{1-\mathbb{E}[B]} + \frac{n^2}{u}\sqrt{\max(\mathbb{V}ar[X_1], .., \mathbb{V}ar[X_{\frac{n}{u}}])}$$

raízes incorretas é menor a $\frac{1}{n^2}$.

Algoritmo de Reconstrução da Chave Secreta RSA

Complexidade do Algoritmo

A Reconstrução da chave secreta sk do criptossistema RSA (u-primo), pode ser feita em **tempo polinomial** $O(n^2)$ com uma probabilidade maior a $1 - \frac{1}{n^2}$, se temos uma porcentagem δ de bits conhecidos em $\tilde{sk}$ maior a $2 - 2^{\frac{u+2}{2u+1}}$.

$$\mathbb{E}[B] = \frac{(2-\delta)^{2u+1}}{2^{u+2}} < 1 \quad \Rightarrow \quad \delta > 2 - 2^{\frac{u+2}{2u+1}}.$$

Algoritmo de Reconstrução da Chave Secreta RSA

Complexidade do Algoritmo

A Reconstrução da chave secreta sk do criptossistema RSA (u-primo), pode ser feita em **tempo polinomial** $O(n^2)$ com uma probabilidade maior a $1 - \frac{1}{n^2}$, se temos uma porcentagem δ de bits conhecidos em $\tilde{sk}$ maior a $2 - 2^{\frac{u+2}{2u+1}}$.

$$\mathbb{E}[B] = \frac{(2-\delta)^{2u+1}}{2^{u+2}} < 1 \quad \Rightarrow \quad \delta > 2 - 2^{\frac{u+2}{2u+1}}.$$

Para Reconstruir a Chave Secreta sk do criptossistema

- RSA básico é preciso um $\delta > 2 - 2^{\frac{4}{5}} = 0.2589$
- RSA 3-primos é preciso um $\delta > 2 - 2^{\frac{5}{7}} = 0.3593$
- RSA 4-primos é preciso um $\delta > 2 - 2^{\frac{6}{9}} = 0.4126$

4 - Implementação

Implementação do algoritmo HS

- O algoritmo de reconstrução foi implementado na linguagem C usando a biblioteca *Relic-toolkit* e testado sob um processador Intel Core I3 2.4 Ghz com 3 Mb de cache e 4 Gb de memória DDR3.
- Os experimentos foram feitos para chaves 2048, 3072 e 4096 bits e para determinados valores de δ.
- Para cada chave de tamanho n foi gerado 100 criptossitemas e para cada criptossistema e cada δ foi gerado 100 chaves secretas com bits modificados.
- Todos os criptossistemas tinham o expoente de encriptação $e = 2^{16} + 1$.
- Os experimentos foram feitos para os criptossistemas RSA onde $2 \leq u \leq 4$.
- Os experimentos foram testados com os valores corretos de $\langle k, k_1, k_2, ..., k_u \rangle$.

Resultados - 2048 bits

Para uma Chave Secreta *sk* RSA básico temos que $\delta > 2 - 2^{\frac{4}{5}} = 0.2589$.

- para $\delta = 0.26$ vamos analisar menos de $48n + 45n^2$ raízes incorretas.
- para $\delta = 0.27$ vamos analisar menos de $5n + 5n^2$ raízes incorretas.

	Quantidade de raízes analisadas			# exp.	Tempo
δ	Mínimo	Máximo	Média	(> 1M)	Média(s)
0.29	2036	571178	4012	0	0.151949
0.28	2283	776810	5060	0	0.191559
0.27	2555	850244	7588	3	0.290930
0.26	2994	977055	14624	10	0.586342
0.25	4029	982756	26879	49	1.061377
0.24	4939	996729	60232	245	2.437966

Resultados - 2048 bits

Para uma Chave Secreta sk RSA 3-primos temos que $\delta > 2 - 2^{\frac{5}{7}} = 0.3593$.

- para $\delta = 0.36$ vamos analisar menos de $59n + 62n^2$ raízes incorretas.
- para $\delta = 0.37$ vamos analisar menos de $4n + 4n^2$ raízes incorretas.

	Quantidade de raízes analisadas			# exp.	Tempo
δ	Mínimo	Máximo	Média	(> 1M)	Média
0.39	1711	147153	4116	0	0.666293
0.38	2140	388317	5638	1	0.927814
0.37	2288	713089	8645	1	1.428584
0.36	2613	928901	13820	14	2.245756
0.35	3878	964553	24107	28	3.987935
0.34	4218	997646	53173	156	8.919880

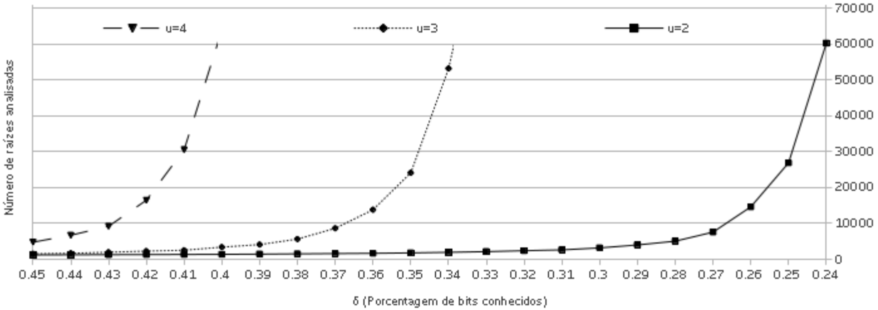
Resultados - 2048 bits

Para uma Chave Secreta *sk* RSA 4-primos temos que $\delta > 2 - 2^{\frac{6}{9}} = 0.4126$.

- para $\delta = 0.42$ vamos analisar menos de $5n + 5n^2$ raízes incorretas.
- para $\delta = 0.43$ vamos analisar menos de $2n + 3n^2$ raízes incorretas.

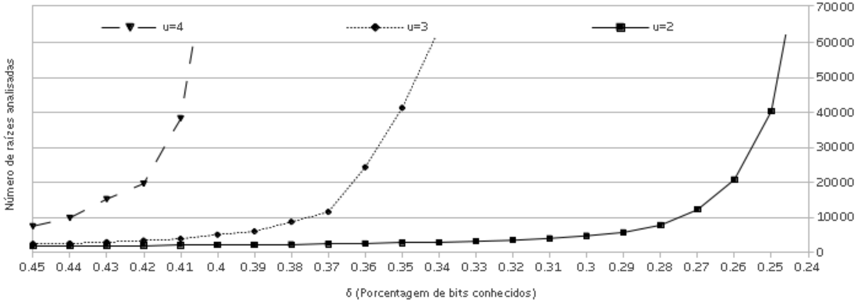
	Quantidade de raízes analisadas			# exp.	Tempo
δ	Mínimo	Máximo	Média	(> 1M)	Média
0.45	1819	110096	4781	0	1.212781
0.44	1884	794452	6714	0	1.692482
0.43	2326	870455	9216	0	2.383156
0.42	2852	618823	16484	3	4.446705
0.41	3802	998132	30423	72	7.928861
0.40	5796	963273	63909	127	16.224300

Resultados da Reconstrução da Chave Secreta RSA 2048



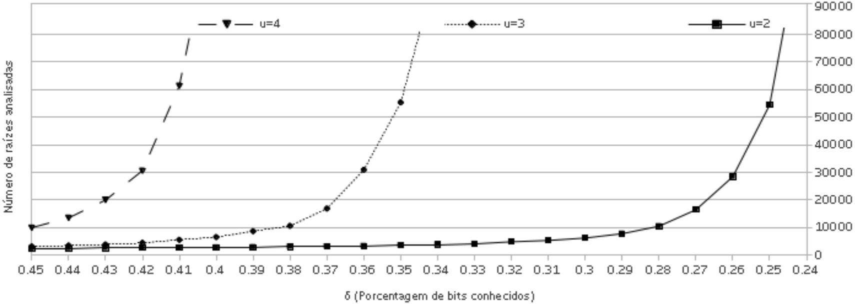
- Para $\delta > 2 - 2^{\frac{u+2}{2u+1}}$ temos um crescimento regular(linear).
- Para $\delta \leq 2 - 2^{\frac{u+2}{2u+1}}$ temos um crescimento exponencial.

Resultados da Reconstrução da Chave Secreta RSA 3072



- Temos um mesmo comportamento para os valores de δ.
- Os resultados estão em proporção $\frac{3072}{2048} = \frac{3}{2}$ com relação aos experimentos feitos para o módulo 2048.

Resultados da Reconstrução da Chave Secreta RSA 4096



- Temos um mesmo comportamento para os valores de δ.
- Os resultados estão em proporção $\frac{4096}{2048} = \frac{2}{1}$ com relação aos experimentos feitos para o módulo 2048.

5 - Conclusões

Nossos Resultados

- É possível reconstruir a chave Secreta sk RSA onde $N = \prod_{i=1}^{u} r_i$ em **tempo polinomial** $O(n^2)$ tendo uma porcentagem $\delta > 2 - 2^{\frac{u+2}{2u+1}}$ de bits conhecidos de $\tilde{sk}$. Alguns resultados:
  - Criptossistema RSA básico ($\delta \geq 0.27$).
  - Criptossistema RSA 3-primos ($\delta \geq 0.37$).
  - Criptossistema RSA 4-primos ($\delta \geq 0.42$).

Nossos Resultados

- É possível reconstruir a chave Secreta sk RSA onde $N = \prod_{i=1}^{u} r_i$ em **tempo polinomial** $O(n^2)$ tendo uma porcentagem $\delta > 2 - 2^{\frac{u+2}{2u+1}}$ de bits conhecidos de $\tilde{sk}$. Alguns resultados:
  - Criptossistema RSA básico ($\delta \geq 0.27$).
  - Criptossistema RSA 3-primos ($\delta \geq 0.37$).
  - Criptossistema RSA 4-primos ($\delta \geq 0.42$).
- A segurança oferecida pelo criptossistema RSA multi-primo sob o RSA básico é maior.

$$2 - 2^{\frac{u+2}{2u+1}} > 2 - 2^{\frac{4}{5}} \approx 0.27 \text{ para } u \geq 3$$

Nossos Resultados

- É possível reconstruir a chave Secreta sk RSA onde $N = \prod_{i=1}^{u} r_i$ em **tempo polinomial** $O(n^2)$ tendo uma porcentagem $\delta > 2 - 2^{\frac{u+2}{2u+1}}$ de bits conhecidos de $\tilde{sk}$. Alguns resultados:
  - Criptossistema RSA básico ($\delta \geq 0.27$).
  - Criptossistema RSA 3-primos ($\delta \geq 0.37$).
  - Criptossistema RSA 4-primos ($\delta \geq 0.42$).
- A segurança oferecida pelo criptossistema RSA multi-primo sob o RSA básico é maior.

$$2 - 2^{\frac{u+2}{2u+1}} > 2 - 2^{\frac{4}{5}} \approx 0.27 \text{ para } u \geq 3$$

- [Teórico] Limite de segurança oferecida pelo criptossistema RSA u-primos.

$$\lim_{u \to \infty} 2 - 2^{\frac{u+2}{2u+1}} \approx 0.58583$$

Quadro de comparações do RSA básico com suas variantes

Criptossistema RSA				
RSA multi-potência[4]		RSA básico		RSA multi-primo
$N = r_1{}^m r_2$ $(m \geq 2)$		$N = r_1 r_2$		$N = \prod_{i=1}^{u} r_i$ $(u \geq 3)$
Número de soluções para $\langle k, k_1, ..., k_u \rangle$ sem conhecer o valor de k				
$\alpha(2)$	$\approx$	$\alpha(2)$	$<$	$\alpha(u)$
Número de bits mais significativos de $\tilde{d}$ que são precisos para calcular k				
$n/(m+1)$	$>$	$n/2$	$<$	n/u
Número de soluções para $\langle k_1, ..., k_u \rangle$ conhecendo o valor de k				
2	$\approx$	$\beta(2) \leq 2$	$<$	$\beta(u) \leq 2(e-2)^{u-2}$
Reconstrução da Chave Secreta RSA em tempo polinomial				
$\delta \geq 2 - 2^{\frac{4}{5}} \approx 0.27$	$\approx$	$\delta \geq 2 - 2^{\frac{4}{5}} \approx 0.27$	$<$	$\delta > 2 - 2^{\frac{u+2}{2u+1}}$

[4] Desenvolvido por Jun Kogure em 2012

Trabalhos Futuros - Problemas Abertos

- **Fatoração um inteiro $N = r_1 r_2$ só tendo bits aleatórios de r_1**
  - Fatoração um inteiro N u-primos só tendo bits aleatórios de $u - 1$ ou menos primos.

Trabalhos Futuros - Problemas Abertos

- **Fatoração um inteiro $N = r_1 r_2$ só tendo bits aleatórios de r_1**
  - Fatoração um inteiro N u-primos só tendo bits aleatórios de $u - 1$ ou menos primos.
- **Diminuição da complexidade do algoritmo de reconstrução**
  - reticulados (*lattice reduction*)

Trabalhos Futuros - Problemas Abertos

- **Fatoração um inteiro $N = r_1 r_2$ só tendo bits aleatórios de r_1**
  - Fatoração um inteiro N u-primos só tendo bits aleatórios de $u-1$ ou menos primos.
- **Diminuição da complexidade do algoritmo de reconstrução**
  - reticulados (*lattice reduction*)
- **Utilização de toda a chave secreta**

$$\tilde{sk}\langle \tilde{d}, \tilde{r_1}, \tilde{r_2}, \tilde{d_1}, \tilde{d_2}, \tilde{r_2}^{-1}, \langle \tilde{r_3}, \tilde{d_3}, \tilde{t_3} \rangle, ..., \langle \tilde{r_u}, \tilde{d_u}, \tilde{t_u} \rangle \rangle$$

para reconstrução e obtenção de sk.

Obrigado!!!

D. F. Aranha e C. P. L. Gouvêa.
RELIC is an Efficient LIbrary for Cryptography.
http://code.google.com/p/relic-toolkit/.

Dan Boneh, Glenn Durfee e Yair Frankel.
An attack on rsa given a small fraction of the private key bits.
Em Advances in Cryptology?ASIACRYPT?98, páginas 25–34. Springer, 1998.

H. Bar-El.
Introduction to side channel attacks.
White Paper. Discretix Technologies Ltd, 2003.

Binomial theorem.
Disponível em http://en.wikipedia.org/wiki/Binomial_theorem.

Geoffrey Mallin Clarke e Dennis Cooke.
A basic course in statistics.
Arnold New York, 1998.

Scott Contini.
General purpose factoring records.
Disponível em http://www.crypto-world.com/FactorRecords.html.

D. Coppersmith.
Small solutions to polynomial equations, and low exponent rsa vulnerabilities.
Journal of Cryptology, 10(4):233–260, 1997.

Compaq Computer Corporation.

Cryptography using compaq multi-prime technology in a parallel processing environment.
Disponível em ftp://ftp.compaq.com/pub/solutions/CompaqMultiPrimeWP.pdf.

Whitfield Diffie e Martin E. Hellman.
New directions in cryptography, 1976.

Paulo Feofiloff.
Introdução informal à complexidade de problemas.

Gnu lesser general public license.
http://www.gnu.org/licenses/lgpl.html.

Mathias Herrmann e Alexander May.
Solving linear equations modulo divisors: On factoring given any bits.
Em Advances in Cryptology-ASIACRYPT 2008, páginas 406–424. Springer, 2008.

David Hook.
Beginning cryptography and PKI in Java.
2005.

Nadia Heninger e Hovav Shacham.
Reconstructing rsa private keys from random key bits, 2009.

J.A. Halderman, S.D. Schoen, N. Heninger, W. Clarkson, W. Paul, J.A. Calandrino, A.J. Feldman, J. Appelbaum e E.W. Felten.
Lest we remember: cold-boot attacks on encryption keys.
Communications of the ACM, 52(5):91–98, 2009.

J. Jonsson e B. Kaliski.
Public-key cryptography standards (pkcs)# 1: Rsa cryptography specifications version 2.1.
Relatório técnico, RFC 3447, February, 2003.

Jun Kogure, Noboru Kunihiro e Hirosuke Yamamoto.
Generalized security analysis of the random key bits leakage attack.
Em Information Security Applications, páginas 13–27. Springer, 2012.

J. Katz e Y. Lindell.
Introduction to modern cryptography.
Chapman & Hall/CRC cryptography and network security. Chapman & Hall/CRC, 2008.

RSA Labs.
Public-key cryptography standards (pkcs), 1991.
Disponível em http://www.rsa.com/rsalabs/node.asp?id=2124.

RSA Labs.
Syslinux project, 1991.
Disponível em http://www.syslinux.org/wiki/index.php/The_Syslinux_Project.

Arjen K Lenstra.
Unbelievable security matching aes security using public key systems.
Em Advances in Cryptology?ASIACRYPT 2001, páginas 67–86. Springer, 2001.

U.M. Maurer.
Factoring with an oracle.

Em Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques, páginas 429–436. Springer-Verlag, 1992.

S. Maitra, S. Sarkar e S. Sen Gupta.
Publishing upper half of rsa decryption exponent.
Advances in Information and Computer Security, páginas 25–39, 2010.

Pearson product moment correlation coefficient.
Disponível em
`http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient`.

Colin Percival.
Cache missing for fun and profit.
Em Proc. of BSDCan 2005, 2005.

J.J. Quisquater e C. Couvreur.
Fast decipherment algorithm for rsa public-key cryptosystem.
Electronics letters, 18(21):905–907, 1982.

M.Y. Rhee.
Cryptography and secure communications.
McGraw-Hill series on computer communications. McGraw-Hill, 1994.

R.L. Rivest, A. Shamir e L. Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
Communications of the ACM, 21(2):120–126, 1978.

S. Skorobogatov.

Low temperature data remanence in static ram.
University of Cambridge Computer Laborary Technical Report, 536, 2002.

Adi Shamir e Nicko Van Someren.
Playing hide and seek with stored keys.
Em Lecture Notes in Computer Science, páginas 118–124, 1998.

R. Terada.
Segurança de dados: criptografia em redes de computador.
Edgard Blucher, 2000.

Laboratório de Segurança de Dados - IME.
Factoring the multi-prime modulus n with random bits.
Artigo a ser submetido.

Paul Zimmermann.
Integer factoring records.
Disponível em http://www.loria.fr/~zimmerma/records/factor.html.